The random component-wise power method

Oguzhan Teke, Palghat P. Vaidyanathan

Oguzhan Teke, Palghat P. Vaidyanathan, "The random component-wise power method," Proc. SPIE 11138, Wavelets and Sparsity XVIII, 111381L (9 September 2019); doi: 10.1117/12.2530511

SPIE.

Event: SPIE Optical Engineering + Applications, 2019, San Diego, California, United States

The Random Component-Wise Power Method

Oguzhan Teke and Palghat P. Vaidyanathan

California Institute of Technology, 1200 E. California Blvd., Pasadena, USA

ABSTRACT

This paper considers a random component-wise variant of the unnormalized power method, which is similar to the regular power iteration except that only a random subset of indices is updated in each iteration. For the case of normal matrices, it was previously shown that random component-wise updates converge in the mean-squared sense to an eigenvector of eigenvalue 1 of the underlying matrix even in the case of the matrix having spectral radius larger than unity. In addition to the enlarged convergence regions, this study shows that the eigenvalue gap does not directly affect the convergence rate of the randomized updates unlike the regular power method. In particular, it is shown that the rate of convergence is affected by the phase of the eigenvalues in the case of random component-wise updates, and the randomized updates favor negative eigenvalues over positive ones. As an application, this study considers a reformulation of the component-wise updates revealing a randomized algorithm that is proven to converge to the dominant left and right singular vectors of a normalized data matrix. The algorithm is also extended to handle large-scale distributed data when computing an arbitrary rank approximation of an arbitrary data matrix. Numerical simulations verify the convergence of the proposed algorithms under different parameter settings.

Keywords: Fixed point iteration, randomized iterations, low-rank approximation, distributed computation.

1. INTRODUCTION

Computation of the dominant eigenvector of a matrix is an important problem with vast number of applications in data processing. Due to its fundamental significance there is a rich literature on the numerical techniques that can extract the dominant eigenvectors.[1–5] The power method is one of the oldest techniques which is still used today. One of its most prominent applications is the well-known PageRank algorithm.[6] Given a square matrix $\mathbf{A}$ of size $N \times N$, the *unnormalized* power method is defined through the following recurrence relation:

$$\mathbf{x}_k = \mathbf{A}\,\mathbf{x}_{k\text{-}1}, \tag{1}$$

where $\mathbf{x}_k$ denotes the vector at the k^{th} iteration of the method. Iterations defined through the recurrence relation in (1) are known to converge to the dominant eigenvector of the given matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$ when the eigenvalues of $\mathbf{A}$ satisfy $|\lambda| \leqslant 1$ (or, if the vector $\mathbf{x}_k$ is properly normalized after each iteration).

In this study we will consider a random and component-wise variant of the regular power method (1), in which we select a random subset of rows of $\mathbf{A}$ in each iteration, compute the inner products between the rows and the vector $\mathbf{x}_{k\text{-}1}$, and then update only the corresponding indices of the vector $\mathbf{x}_{k\text{-}1}$. The values of the unselected indices remain unchanged. This randomized variant is first proposed in order to study the behavior of autonomous networks[7–9] in the context of graph signal processing. Here we will consider the randomized update scheme from a purely linear algebraic point of view.

Regarding the randomized variant considered here we ask the following questions: Do the iterations converge? If so, where and how fast do they converge? When $\mathbf{A}$ is a normal matrix, under mild boundedness assumptions on the matrix $\mathbf{A}$, it is proven that iterations indeed converge in the mean-squared sense to an eigenvector of $\mathbf{A}$ with eigenvalue 1.[7] However, convergence characteristics of the random updates differ significantly from that of the regular power method in two ways. The first one is the region of convergence: random component-wise

Contact e-mails: oteke@caltech.edu, ppvnath@systems.caltech.edu. This work was supported in parts by the ONR grant N00014-18-1-2390, the NSF grant CCF-1712633, and the Electrical Engineering Carver Mead Research Seed Fund of the California Institute of Technology.

Wavelets and Sparsity XVIII, edited by Dimitri Van De Ville, Manos Papadakis, Yue M. Lu, Proc. of SPIE
Vol. 11138, 111381L · © 2019 SPIE · CCC code: 0277-786X/19/$21 · doi: 10.1117/12.2530511

updates have a larger region of convergence on the eigenvalue plane. Thus, randomized updates are guaranteed to converge whenever the regular power method converges. More importantly, randomized updates can converge even when the regular power method does not.

In addition to the enlarged convergence regions,[7] in this study we consider the effect of the randomized updates on the rate of convergence. In this regard, we first demonstrate that randomized updates may converge *faster*, or slower than the regular power method depending on the sign (or, phase) of the non-unit eigenvalues, and the eigenvalue gap does not have a direct effect on the convergence rate unlike the regular power method. Then, we explain the intricate relationship between the rate of convergence and the eigenvalues of $\mathbf{A}$, and describe the cases for which randomized updates are faster, or slower than the standard power iteration.

In this study, we also reformulate the random component-wise updates in order to compute the dominant singular vectors of an arbitrary data matrix. In particular, we propose an algorithm that is proven to converge to the dominant left and right singular vectors of a normalized data matrix. Then, we extend the algorithm to compute an arbitrary rank approximation of an arbitrary matrix. Due to its asynchronous and component-wise nature, we further extend the algorithm for a distributed computation. More precisely, the data is assumed to be partitioned into smaller pieces, and each piece is stored and processed by a single core. These cores do computations locally and communicate with a fusion center in a randomized and asynchronous manner.

1.1 Previous Literature

Due to the component-wise nature, the update scheme considered here resembles the coordinate descent type algorithms.[10–13] In particular, the approach by Lei, et al.[14] considers the same update scheme in which the update sets are selected deterministically and adaptively in each iteration in order to improve the rate of convergence.

In this study, the random component-wise updates will be analyzed from the view-point of the asynchronous linear fixed point iterations (state recursions). We note that non-random versions of asynchronous (possibly non-linear) fixed point iterations are well studied in the literature.[15–17] For the linear model considered in this study, the earliest study by Chazan and Miranker provided the necessary and sufficient condition under which the asynchronous iterations are guaranteed to converge for any index sequence.[15] More recent studies[18,19] considered the randomized variations of asynchronous iterations, in which indices are assumed to be selected with equal probabilities, and they provided sufficiency conditions for the convergence.

In this study, we consider a stochastic model in which a *random subset* of indices is updated in each iteration. Assuming that $\mathbf{A}$ is a normal matrix, we show that the iterations converge to a fixed point of $\mathbf{A}$, i.e., an eigenvector of eigenvalue 1, in the mean-squared sense even when the matrix has other eigenvalues with magnitudes greater than unity. In addition, based on its convergence properties, we describe the conditions on the eigenvalues of $\mathbf{A}$ under which the random asynchronous updates converge faster (or, slower) than its synchronous counter-part.

1.2 Outline

The paper is organized as follows. In Section 2 we first define the random component-wise power method where we precisely describe the statistical model of the randomness. In Section 2.1 we present an upper bound on the rate of convergence of the random updates. In Section 2.2 we characterize a region for the eigenvalues such that random updates are guaranteed to converge. In Section 2.3 we compare and contrast the random updates with the regular power method in terms of rate of convergence. We specify the cases under which randomized updates converge faster, slower, or at a similar rate. In Section 3 we consider a reformulation of the power method and propose an algorithm that is proven to converge to the left and right dominant singular vectors of a normalized data matrix. In Section 3.1, we extend the proposed algorithm in order to obtain an arbitrary rank approximation of an arbitrary data matrix. In Section 3.2 we leverage the component-wise and asynchronous nature of the proposed algorithm in order to compute the singular vectors of *distributed* data with distributed computations.

1.3 Preliminaries and Notation

We will use $\mathbb{P}[\cdot]$ and $\mathbb{E}[\cdot]$ to denote the probability and expectation, respectively. Positive definite and semi-definite ordering will be denoted by $\prec$ and $\preceq$, respectively. For a matrix $\mathbf{X}$ we will use $\mathbf{X}^{\mathrm{H}}$ to denote its conjugate transpose, use $\mathbf{X}_{[:,i]}$ to denote its i^{th} column, and use $\mathbf{X}_{[i,:]}$ to denote its i^{th} row. We will use $\mathrm{diag}(\mathbf{X})$ to denote the diagonal masking of $\mathbf{X}$, that is, $(\mathrm{diag}(\mathbf{X}))_{i,i} = X_{i,i}$ and $(\mathrm{diag}(\mathbf{X}))_{i,j} = 0$ for $i \neq j$. We will use $\mathcal{T}$ to denote a subset of $\{1, \cdots, N\}$, and its size is denoted as $|\mathcal{T}|$.

2. RANDOM COMPONENT-WISE POWER METHOD

In this study we will consider a random component-wise variation of (1) and always assume that $\mathbf{A}$ is a normal matrix, i.e., $\mathbf{A}\,\mathbf{A}^{\mathrm{H}} = \mathbf{A}^{\mathrm{H}}\,\mathbf{A}$. More precisely, we consider the following type of recurrence relation:

$$(\mathbf{x}_k)_i = \begin{cases} (\mathbf{A}\,\mathbf{x}_{k\text{-}1})_i, & i \in \mathcal{T}_k, \\ (\mathbf{x}_{k\text{-}1})_i, & i \notin \mathcal{T}_k, \end{cases} \tag{2}$$

where $\mathcal{T}_k \subseteq \{1, \cdots, N\}$ denotes a randomly selected set of indices at the k^{th} iteration. In words, the k^{th} iteration of the scheme in (2) first computes the regular power iteration $\mathbf{A}\,\mathbf{x}_{k\text{-}1}$, but it updates only the indices specified by the random subset $\mathcal{T}_k$. The values of the indices not belonging to $\mathcal{T}_k$ remain unchanged.

When the matrix $\mathbf{A}$ is considered as a local graph operator, i.e., the adjacency matrix, or the graph Laplacian, the scheme in (2) models the random asynchronous behavior of the nodes of a graph.[7,8] In this setting, the random asynchronous model (2) allows us to design polynomial graph filters that result in clustering algorithms for autonomous networks.[7,8] When extended to have a constant input signal, the model (2) is also useful for a node-asynchronous implementation of rational filters on graphs.[9,20]

When $\mathbf{A}$ is viewed as a data matrix, which is the case to be considered in this study, the component-wise updates of (2) allow distributed computation of the dominant eigenvector since the rows of $\mathbf{A}$ need not be accessed simultaneously. That is, the data matrix $\mathbf{A}$ can be partitioned (row-wise) into different agents, and each agent uses only its own partition to update the corresponding indices of the shared memory. Due to the point-wise update structure of (2), the agents are allowed to operate randomly and asynchronously. We will elaborate on this idea later in Section 3.

It is assumed that the update set $\mathcal{T}_k$ is selected randomly and independently among all possible 2^N different subsets of $\{1, \cdots, N\}$ at every iteration of (2). In this study, we consider the following stochastic model for the selection of the subsets. There are two sources of randomness: 1) *The size* of the set $\mathcal{T}_k$ is selected randomly. 2) Once the size is determined, *the content* of the subset is selected uniformly randomly among all subsets of the selected size. More precisely, the probability of $\mathcal{T}_k$ being equal to a specific subset $\mathcal{T}$ is given as follows:

$$\mathbb{P}[\,\mathcal{T}_k = \mathcal{T}\,] = p_t \binom{N}{t}^{-1}, \qquad \text{where} \qquad t = |\mathcal{T}|, \tag{3}$$

where p_t denotes the probability of $\mathcal{T}_k$ being of size t, i.e., $p_t = \mathbb{P}[\,|\mathcal{T}_k| = t\,]$ for $1 \leqslant t \leqslant N$. By adjusting the values of p_t's, the stochastic model in (3) can capture different scenarios. For example, the case of $p_1 = 1$ implies that only one index is selected uniformly at random at every iteration. The case of $p_N = 1$ implies that $\mathcal{T}_k$ is selected to be $\{1, \cdots, N\}$ with probability 1, which corresponds to the regular power method in (1). For the general case, let T denote the size of the set $\mathcal{T}_k$, i.e. $\mathsf{T} = |\mathcal{T}_k|$. Thus, T is a discrete random variable whose distribution (specified by p_t's) will be shown to determine the convergence characteristics of the random component-wise power method. More specifically, we define the following quantity:

$$\delta_{\mathsf{T}} = \frac{\mathbb{E}[\,\mathsf{T}\,(N-\mathsf{T})\,]}{\mathbb{E}[\,\mathsf{T}\,(N-1)\,]} = \frac{N - \mu_{\mathsf{T}} - \sigma_{\mathsf{T}}^2/\mu_{\mathsf{T}}}{N-1}, \tag{4}$$

where μ_{T} and σ_{T}^2 denote the mean and the variance of T, respectively. Notice that $0 \leqslant \delta_{\mathsf{T}} \leqslant 1$ is always satisfied. Furthermore, it is readily verified that

$$\delta_{\mathsf{T}} = 1 \quad \Longleftrightarrow \quad \mu_{\mathsf{T}} = 1, \qquad \text{and} \qquad \delta_{\mathsf{T}} = 0 \quad \Longleftrightarrow \quad \mu_{\mathsf{T}} = N. \tag{5}$$

That is, $\delta_{\mathsf{T}} = 0$ if and only if all the indices are selected at every iteration (regular power method), which requires all the nodes to be updated simultaneously. On the other hand, $\delta_{\mathsf{T}} = 1$ if and only if exactly one index is selected at every iteration, in which case no synchronization is required between consecutive updates. Based on this observation, δ_{T} is referred to as *the amount of asynchronicity* of the updates.[7] As we shall see in the next section, the quantity δ_{T} plays a key role in the convergence characteristics of the random updates.

2.1 Convergence in the Mean-Squared Sense

We start by noting that the convergence properties of the model (2) were presented in a recent study.[7] Nevertheless, we will summarize some of these results here in order to keep this study self-contained.

Since the matrix $\mathbf{A}$ is assumed to be normal, it has an orthonormal set of eigenvectors, but eigenvalues of $\mathbf{A}$ need not be real valued. In order to ensure that the update model (2) has a fixed point, the matrix $\mathbf{A}$ is required to have eigenvalue 1 (unit eigenvalue),[7] and the eigenvalue 1 can have multiplicity $m \geqslant 1$. We note that the need for the unit eigenvalue can be eliminated when a normalized step is introduced to the update scheme of (2), which will be elaborated later in Section 3.

Without loss of generality the eigenvalues of $\mathbf{A}$ can be enumerated such that $\lambda_i \neq 1$ for $1 \leqslant i \leqslant N\text{-}m$, and $\lambda_i = 1$ for $i > N\text{-}m$. It is important to note that non-unit eigenvalues (λ_i for $1 \leqslant i \leqslant N\text{-}m$) are not assumed be strictly inside the unit circle: *their magnitude may be larger than unity.* Under these assumptions, we can write the eigenvalue decomposition of $\mathbf{A}$ as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{U} & \mathbf{V}_1 \end{bmatrix} \begin{bmatrix} \lambda_1 & & & & & \\ & \ddots & & & & \\ & & \lambda_{N\text{-}m} & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} \mathbf{U} & \mathbf{V}_1 \end{bmatrix}^{\mathrm{H}}, \tag{6}$$

where $\mathbf{V}_1 \in \mathbb{C}^{N \times m}$ is an orthonormal basis for the eigenspace of the unit eigenvalue, and $\mathbf{U} \in \mathbb{C}^{N \times (N\text{-}m)}$ corresponds to the eigenvectors of the non-unit eigenvalues. Since $\mathbf{A}$ is assumed to be a normal matrix, we have $\mathbf{U}^{\mathrm{H}}\,\mathbf{V}_1 = \mathbf{0}$, and $\mathbf{U}^{\mathrm{H}}\,\mathbf{U} = \mathbf{I}$. We now define the following quantity:

$$\rho = \lambda_{\max}\Big(\mathbf{U}^{\mathrm{H}}\ \mathrm{diag}\left(\mathbf{U}\,\mathbf{U}^{\mathrm{H}}\right)\mathbf{U}\Big), \tag{7}$$

where $\mathrm{diag}(\cdot)$ denotes the diagonal masking operator for matrices. Notice that only the column space of $\mathbf{U}$ determines the value of ρ. More importantly, we have following property (whose proof is provided in Appendix A):

LEMMA 1. *The following holds true for any* $\mathbf{U} \in \mathbb{C}^{N \times (N\text{-}m)}$ *with* $\mathbf{U}^{\mathrm{H}}\,\mathbf{U} = \mathbf{I}$*:*

$$\frac{1}{N}\,\mathbf{I} \ \preceq\ \mathbf{U}^{\mathrm{H}}\ \mathrm{diag}\left(\mathbf{U}\,\mathbf{U}^{\mathrm{H}}\right)\mathbf{U} \ \preceq\ \mathbf{I}. \tag{8}$$

We note that a looser version of the inequality (8) (which shows that $\mathbf{0} \prec \mathbf{U}^{\mathrm{H}}\,\mathrm{diag}(\mathbf{U}\,\mathbf{U}^{\mathrm{H}})\mathbf{U} \preceq \mathbf{I}$ only) was mentioned in previous studies without a proof.[7,21] However, this study presents the inequality (8) with a formal proof. Furthermore, Lemma 1 implies the following inequality regarding the quantity ρ:

$$\frac{1}{N} \leqslant \rho \leqslant 1. \tag{9}$$

Since the eigenspace $\mathbf{V}_1$ contains the fixed points of the update scheme of (2), and the iterand $\mathbf{x}_k$ is expected to converge to a fixed-point of the model (2) we define the residual vector $\mathbf{r}_k$ at the k^{th} iteration as follows:

$$\mathbf{r}_k = \mathbf{x}_k - \mathbf{V}_1\,\mathbf{V}_1^{\mathrm{H}}\,\mathbf{x}_k = \mathbf{U}\,\mathbf{U}^{\mathrm{H}}\,\mathbf{x}_k, \tag{10}$$

which is the residual from the projection of $\mathbf{x}_k$ onto the column space of $\mathbf{V}_1$. Notice that the convergence of $\mathbf{x}_k$ to an eigenvector of the unit eigenvalue (a fixed point) is equivalent to the convergence of $\mathbf{r}_k$ to zero. The following theorem (whose proof is presented in an earlier study[7]) provides an upper bound for $\mathbf{r}_k$ as follows:

THEOREM 1. *The expected squared ℓ_2-norm of the residual at the k^{th} iteration is bounded as follows:*

$$\mathbb{E}\big[\,\|\mathbf{r}_k\|_2^2\,\big] \;\leqslant\; \Gamma^k\,\|\mathbf{r}_0\|_2^2, \qquad \text{where} \qquad \Gamma = \max_{1\leqslant j\leqslant N\text{-}m} \; c\,(\lambda_j), \tag{11}$$

and

$$c\,(\lambda) = 1 + \frac{\mu_{\mathsf{T}}}{N}\bigg(\,|\lambda|^2 - 1 + \delta_{\mathsf{T}}\,(\rho-1)\,|\lambda-1|^2\bigg). \tag{12}$$

♢

We first note that the function $c\,(\lambda)$ defined in (12) can be considered as *the cost associated with an eigenvalue* regarding the convergence of the randomized asynchronous updates. Thus, the rate parameter Γ in (11) is determined simply by the maximum cost corresponding to all non-unit eigenvalues.

Unlike the regular power method whose behavior is solely determined by the eigenvalues, Theorem 1 shows that the convergence of random component-wise updates depends also on the eigenspace geometry, ρ, as well as the amount of asynchronicity, μ_{T} and δ_{T}, of the updates. Notice that in the case of $\delta_{\mathsf{T}} = 0$ (which also implies $\mu_{\mathsf{T}} = N$), the cost of an eigenvalue reduces to $c\,(\lambda) = |\lambda|^2$, and the rate parameter reduces to $\Gamma = \max_{1\leqslant j\leqslant N\text{-}m}|\lambda_j|^2$. Thus, Theorem 1 is consistent with the well-known behavior of the regular power iteration.

More importantly, Theorem 1 reveals two key properties of the randomized asynchronous updates: 1) The non-unit eigenvalues of $\mathbf{A}$ need not be inside the unit disk in order to ensure the convergence of the updates. 2) Not just the magnitude, but also the phase of an eigenvalue affect the associated cost, so the eigenvalue gap (which is the key factor determining the rate of convergence of the regular power iteration) is not directly related to the convergence rate of the randomized asynchronous counter-part. We will elaborate on these details in the following sections.

2.2 Larger Convergence Regions

For the case of regular power method it is well known that an arbitrary nonzero initial vector $\mathbf{x}_0$ converges to a nonzero $\mathbf{x}$ if and only if $\mathbf{x}$ satisfies $\mathbf{A}\mathbf{x} = \mathbf{x}$ (i.e., 1 is an eigenvalue of $\mathbf{A}$), and the remaining eigenvalues of $\mathbf{A}$ satisfy $|\lambda| < 1$. If there is another eigenvalue satisfying $|\lambda| = 1$, then the vector $\mathbf{x}_k$ may fall into limit cycles, and if $|\lambda| > 1$, the vector grows in an unbounded manner through the iterations. On the contrary, random component-wise updates have different convergence behavior. The following corollary characterizes a region for the non-unit eigenvalues of the matrix $\mathbf{A}$ such that the residual defined in (10) converges to zero in the mean-squared sense as the iterations progress.[7]

COROLLARY 1. *Assume that all non-unit eigenvalues of $\mathbf{A}$ satisfy the following condition:*

$$\left|\lambda - \frac{\alpha}{\alpha+1}\right| < \frac{1}{\alpha+1}, \qquad \textit{where} \qquad \alpha = \delta_{\mathsf{T}}\,(\rho-1). \tag{13}$$

Then,

$$\lim_{k\to\infty} \quad \mathbb{E}\big[\,\|\mathbf{r}_k\|_2^2\,\big] = 0. \tag{14}$$

♢

The convergence region given in (13) defines a disk on the complex plane centered at $\alpha/(\alpha{+}1)$ with radius $1/(\alpha{+}1)$, which is visualized in Figure 1 below. We point out that $1/N \leqslant (\alpha+1) \leqslant 1$ is always satisfied due to the fact that $1/N \leqslant \rho \leqslant 1$ (Lemma 1) and that $0 \leqslant \delta_{\mathsf{T}} \leqslant 1$. Thus, the region defined in (13) always has a finite area, and *it always contains the open unit disk.* It is important to note that the convergence region gets larger as α approaches -1, and it is the smallest when $\alpha = 0$, in which case it becomes the open unit disk itself.

Corollary 1 reveals the interplay between the amount of asynchronicity and the eigenspace geometry, and their effect on the region of convergence. In the case of regular power iteration we have $\delta_{\mathsf{T}} = 0$, thus, $\alpha = 0$ irrespective

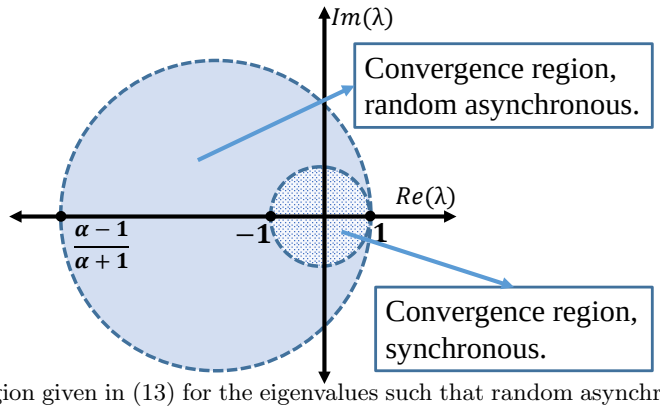


Figure 1. The convergence region given in (13) for the eigenvalues such that random asynchronous updates are guaranteed to converge.

of the eigenspace geometry. As a result, the convergence region of the eigenvalues is the well-known open unit disk. As the updates become more asynchronous (less number of nodes are updated per iteration), the value of α can decrease (i.e., become negative) but cannot increase (since $\rho \leqslant 1$). Thus, the region of convergence never gets smaller. Then, we conclude that random component-wise power iterations are guaranteed to converge whenever the regular power method converges. Furthermore, if $\rho < 1$ (which is the case in most practical applications), then the region of convergence is the largest when $\delta_{\mathsf{T}} = 1$. That is, *updating exactly one index per iteration maximizes the region of convergence*. As a result, random component-wise updates can converge even when the regular power method diverges (depending on the eigenspace geometry).[7] Furthermore, the random component-wise updates do not necessarily converge to the dominant eigenvector; rather they converge to an eigenvector of the eigenvalue 1 (a fixed-point) even when there are eigenvalues with magnitudes larger than unity.

2.3 Rate of Convergence

Although the region of convergence can only be expanded by random component-wise updates as explained in the previous subsection, the rate of convergence has a more intricate behavior that requires a detailed discussion. In the regular power method, the rate of convergence is determined by the eigenvalue gap, which is the difference between the *magnitudes* of the two largest eigenvalues of the matrix $\mathbf{A}$. Thus, the sign (or, the phase in the complex case) of the eigenvalues are not important. Unlike the regular power method, sign of the eigenvalues do matter in the random component-wise updates. As a result, the eigenvalue spectrum has an asymmetric impact on the rate of convergence.

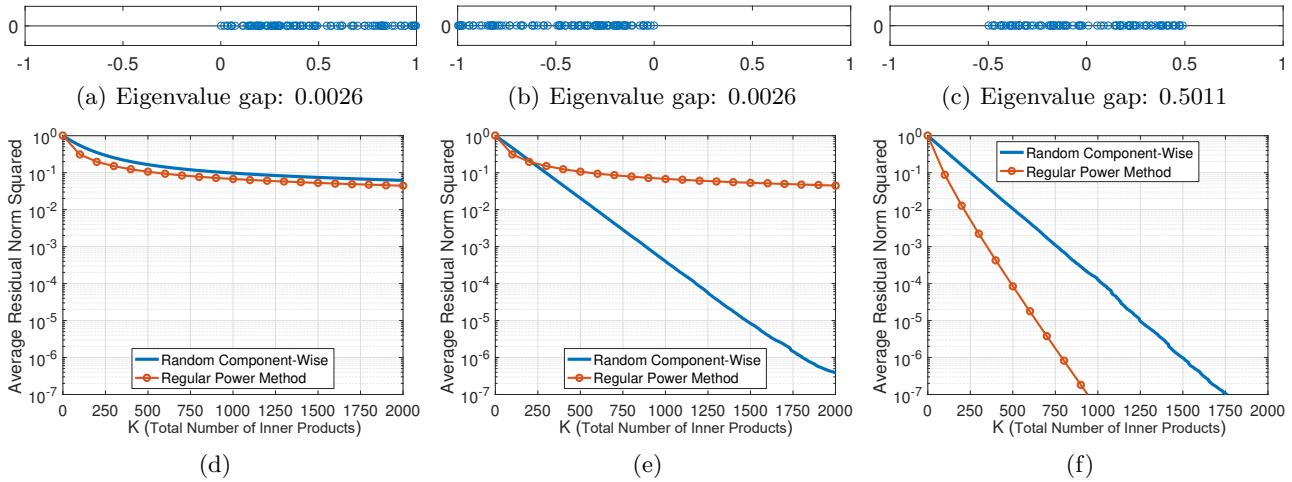
In order to explain the difference between the random component-wise and regular power methods in terms of the rate of convergence, we will first present a numerical example that summarizes our key observations. Then, we will explain the effect of the asynchronicity on the rate of convergence theoretically..

2.3.1 Numerical Observations

In this simulation, random component-wise updates select exactly one index per iteration ($\delta_{\mathsf{T}} = 1$), in which case a single inner product is computed per iteration. On the contrary, the regular power method computes N inner products per iteration. For a fair comparison between the two, we fix *the total number of inner product*, which will be denoted by K. Thus, the regular power method will run $\lceil K/N \rceil$ iterations, whereas the component-wise variant will run K iterations.

For the numerical experiment we consider three symmetric matrices of size $N = 100$. All three matrices are constructed such that $\lambda_N = 1$ is an eigenvalue with multiplicity $m = 1$, and the remaining N-1 eigenvalues are selected to be $|\lambda_i| < 1$ so that the power method (hence any random variant) is guaranteed to converge to an eigenvector of the eigenvalue $\lambda_N = 1$. (See Corollary 1.) In the first two examples we consider a pair of simultaneously diagonalizable matrices. The non-unit eigenvalues of the first matrix are selected to be positive (visualized in Figure 2(a)), and the non-unit eigenvalues of the second matrix are selected to be the negative of those of the first matrix (visualized in Figure 2(b)). In the third example we take a random symmetric matrix

with non-unit eigenvalues satisfying $-0.5 < \lambda_i < 0.5$ (visualized in Figure 2(c)). Figures 2(d), 2(e) and 2(f) show the value of $\mathbb{E}[\|\mathbf{r}_K\|_2^2]/\|\mathbf{r}_0\|_2^2$ with respect to K for the three matrices described above.



(a) Eigenvalue gap: 0.0026 (b) Eigenvalue gap: 0.0026 (c) Eigenvalue gap: 0.5011

(d) (e) (f)

Figure 2. Non-unit eigenvalues of the (a) first, (b) second, (c) third examples. Eigenvalue gap is defined as the difference between 1 and the magnitude of the largest non-unit eigenvalue. Normalized residual errors in the (d) first, (e) second, (f) third examples. Since the regular power method requires N inner products per iteration, the residual error appears only at integer multiples of $N = 100$. Results are obtained by averaging over 10^4 independent runs.

We first compare the results in Figures 2(d) and 2(e). Since the eigenvalues have the same magnitudes, the regular power method behaves the same in both cases. Although the eigenvalue gap is the same in both cases, the random component-wise method converges *significantly faster* when the second dominant eigenvalue is negative. When the second dominant eigenvalue is positive, both the regular and the component-wise updates behave similarly. In the third example, Figure 2(f), the matrix has a large eigenvalue gap, in which case the random component-wise updates do not converge as fast as the regular power method.

2.3.2 Theoretical Justification

In order to explain the behavior in Figure 2, in this section we will assume a slightly simplified stochastic model for the selection of the update sets in (2). Namely, we will assume that the scheme (2) updates *exactly* μ_T indices per iteration. Thus, the random variable T (which denotes the size of the update sets) becomes a deterministic quantity, and $\sigma_\mathsf{T}^2 = 0$. So, the parameter δ_T (the amount of asynchronicity) reduces to the following form:

$$\delta_\mathsf{T} = \frac{N - \mu_\mathsf{T}}{N - 1}. \tag{15}$$

In this setting we note that an update in the form of (2) requires μ_T inner products per iteration. So, the cost of a single power iteration, which requires N inner products, is equivalent to the cost of N/μ_T asynchronous iterations in which μ_T indices are updates simultaneously. Since the associated cost of an eigenvalue defined in (12) disregards the cost of an iteration, we consider the following quantity instead:

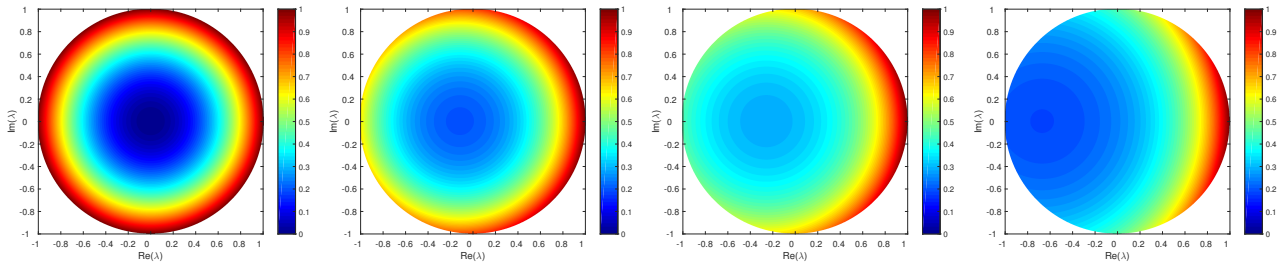
$$r\big(\lambda;\ \mu_\mathsf{T}, \rho\big) = \left(1 + \frac{\mu_\mathsf{T}}{N}\left(|\lambda|^2 - 1 + \delta_\mathsf{T}\,(\rho - 1)\,|\lambda - 1|^2\right)\right)^{N/\mu_\mathsf{T}}, \tag{16}$$

which results in a fair comparison among the component-wise updates with different amount of asynchronicity. The quantity $r\big(\lambda;\ \mu_\mathsf{T}, \rho\big)$ can be interpreted as the amount of reduction in the residual error when eigenvalue λ is present in the matrix $\mathbf{A}$ with the eigenspace parameter ρ, and the model (2) updates μ_T indices simultaneously. Thus, smaller values of $r\big(\lambda;\ \mu_\mathsf{T}, \rho\big)$ indicate a better (faster) convergence of the randomized scheme in (2).

We first note that the quantity $r\big(\lambda;\ \mu_{\mathsf{T}},\ \rho\big)$ can be equivalently re-written as follows:

$$r\big(\lambda;\ \mu_{\mathsf{T}}, \rho\big) = \left(1 + \frac{\mu_{\mathsf{T}}}{N}\ (\alpha+1)\ \left(\left|\lambda - \frac{\alpha}{\alpha+1}\right|^2 - \frac{1}{(\alpha+1)^2}\right)\right)^{N/\mu_{\mathsf{T}}}, \qquad \text{where} \qquad \alpha = \delta_{\mathsf{T}}\ (\rho - 1). \tag{17}$$

Then, it is clear that the point $\lambda^\star = \alpha/(\alpha+1)$ minimizes $r\big(\lambda;\ \mu_{\mathsf{T}},\ \rho\big)$ over the variable λ, and $r\big(\lambda;\ \mu_{\mathsf{T}},\ \rho\big)$ (as a function of λ) is circularly symmetric with respect to the point $\lambda^\star$. In addition, the inequality (9) ensures that $r\big(\lambda;\ \mu_{\mathsf{T}}, \rho\big) \geqslant 0$. In order to demonstrate its behavior, we evaluate $r\big(\lambda;\ \mu_{\mathsf{T}},\ \rho\big)$ numerically over the unit disk (with respect to λ) for different values of μ_{T} and ρ. These computations are visualized in Figure 3.



(a) $r(\lambda;\ N,\ 0.8)$ (b) $r(\lambda;\ N/2,\ 0.8)$ (c) $r(\lambda;\ 1,\ 0.8)$ (d) $r(\lambda;\ 1,\ 0.6)$

Figure 3. Numerical evaluation of $r\big(\lambda, \mu_{\mathsf{T}}\big)$ for various different values of μ_{T} and ρ. The value of N is set to be $N = 100$.

In the case of synchronous updates we have $\mu_{\mathsf{T}} = N$, thus $\alpha = 0$, and the quantity defined in (16) reduces to $r\big(\lambda;\ N,\ \rho\big) = |\lambda|^2$ irrespective of the value of ρ, which can be seen clearly from Figure 3(a). Thus, as $|\lambda|$ approaches 1, the value of $r\big(\lambda;\ N, \rho\big)$ approaches 1 irrespective of the phase of λ. So, only the magnitude of an eigenvalue affects the convergence rate of the regular power iteration, which is a well-known result.

In the case of asynchronous updates we have $\mu_{\mathsf{T}} < N$, and we will assume $\rho < 1$ (which is the case in most practical applications). Thus, we have $\alpha < 0$, and the phase of an eigenvalue becomes important since $r\big(\lambda;\ \mu_{\mathsf{T}},\ \rho\big)$ is no longer a circularly symmetric function of λ with respect to the origin. Figures 3(b), 3(c) and 3(d) visualize this behavior clearly. In particular, note that as λ approaches 1, the quantity $r\big(\lambda;\ \mu_{\mathsf{T}},\ \rho\big)$ approaches 1 as well. On the other hand, as λ approaches -1, the quantity $r\big(\lambda;\ \mu_{\mathsf{T}},\ \rho\big)$ stays bounded away from 1. More precisely,

$$r\big(1;\ \mu_{\mathsf{T}}, \rho\big) = 1, \qquad\qquad r\big(-1;\ \mu_{\mathsf{T}}, \rho\big) = \left(1 + \frac{\mu_{\mathsf{T}}}{N}\, 4\,\alpha\right)^{N/\mu_{\mathsf{T}}}. \tag{18}$$

So, eigenvalues that are close to 1 result in a slower convergence, whereas eigenvalues can be arbitrarily close to -1, yet the convergence does not necessarily slow down. Therefore, in light of (18) and Figure 3 we can conclude that *the random component-wise updates favor negative eigenvalues over positive ones.* This conclusion is consistent with the numerical observations made in Figures 2(d) and 2(e): when the second dominant eigenvalue is close to 1, both the random component-wise updates and the regular power iteration converge slowly. On the contrary, when the second dominant eigenvalue is close to -1, the random component-wise updates converge faster than the synchronous (regular) counter-part. In fact, it is possible to construct a matrix $\mathbf{A}$ (by placing the second dominant eigenvalue sufficiently close to -1) such that the randomized updates converge *arbitrarily faster* than the regular power iteration.

Although random component-wise updates converge faster when the second dominant eigenvalue is close to -1, Figure 2(f) shows that randomized updates are not always faster than the synchronous counter-part. In order to explain the behavior observed in Figure 2(f), we consider $r\big(\lambda;\ \mu_{\mathsf{T}}, \rho\big)$ evaluated at $\lambda = 0$. More precisely,

$$r\big(0;\ \mu_{\mathsf{T}}, \rho\big) = \left(1 + \frac{\mu_{\mathsf{T}}}{N}\Big(\delta_{\mathsf{T}}\ (\rho - 1) - 1\Big)\right)^{N/\mu_{\mathsf{T}}} \geqslant \left(1 - \frac{\mu_{\mathsf{T}}}{N}\right)^{2N/\mu_{\mathsf{T}}}, \tag{19}$$

where the lower bound follows from (9). As long as the updates are randomized (the case of $\mu_{\mathsf{T}} < N$), it is clear from (19) that $r\big(0;\ \mu_{\mathsf{T}}, \rho\big)$ is bounded away from zero. Figures 3(b), 3(c) and 3(d) visualize this behavior as well. Then, we can conclude that *in the case of random component-wise updates the associated cost of an eigenvalue is bounded away from zero even when the eigenvalue itself is close to zero.* This conclusion is consistent with the simulation results presented in Figure 2(f). When the non-unit eigenvalues are close to zero, regular power iteration converges faster than its randomized variant.

As a concluding remark, we note that the results presented in this section are valid when $\mathbf{A}$ is a normal matrix, i.e, $\mathbf{A}$ is unitarily diagonalizable. The results of this section may not hold true when $\mathbf{A}$ is an arbitrary matrix. Nevertheless, the normality condition is not a loss of generality if our goal is to construct a random component-wise method that can compute the singular vectors of an arbitrary matrix. We elaborate on this in the next section.

3. RANDOMIZED COMPUTATION OF THE DOMINANT SINGULAR VECTORS

In most of the data related applications $\mathbf{A}$ happens to be a rectangular matrix, and procedures such as principal component analysis (PCA) require the *singular vectors* of the matrix $\mathbf{A} \in \mathbb{C}^{M\times N}$. Due to the importance of the singular vectors, many efforts have been made to develop fast algorithms, especially randomized ones.[22–26] Although the random component-wise update considered in (2) is not directly applicable to rectangular matrices, a reformulation allows us to compute the dominant singular vectors with random component-wise updates. In this regard, we start by assuming that $M \leqslant N$ without loss of generality, and then consider the *Hermitian dilation* of the given matrix $\mathbf{A}$, which is defined as follows:

$$\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^{\mathrm{H}} & \mathbf{0} \end{bmatrix} \in \mathbb{C}^{(M+N)\times(M+N)}. \tag{20}$$

It is clear that $\bar{\mathbf{A}}$ is an Hermitian matrix, i.e., $\bar{\mathbf{A}}^{\mathrm{H}} = \bar{\mathbf{A}}$. More importantly, the eigenvalue decomposition of $\bar{\mathbf{A}}$ is closely related to the singular value decomposition of $\mathbf{A}$. More precisely, let $\mathbf{A}$ have the following reduced form singular value decomposition:

$$\mathbf{A} = \widetilde{\mathbf{U}}\,\mathbf{\Sigma}\,\widetilde{\mathbf{V}}^{\mathrm{H}} \qquad \text{where} \qquad \widetilde{\mathbf{U}} \in \mathbb{C}^{M\times M}, \quad \mathbf{\Sigma} \in \mathbb{R}^{M\times M}, \quad \widetilde{\mathbf{V}} \in \mathbb{C}^{N\times M} \tag{21}$$

where $\widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{V}}$ are the matrices consisting of the left and right singular vectors of $\mathbf{A}$, respectively, and $\mathbf{\Sigma}$ is the diagonal matrix consisting of the singular values of $\mathbf{A}$, i.e., $\Sigma_{i,i} = \sigma_i$. We assume that the singular values are in the descending order, i.e., $\sigma_1 \geqslant \cdots \geqslant \sigma_M$. Then, the eigenvalue decomposition of $\bar{\mathbf{A}}$ is as follows:

$$\bar{\mathbf{A}} = \mathbf{Q}\,\mathbf{\Lambda}\,\mathbf{Q}^{\mathrm{H}}, \qquad \text{where} \qquad \mathbf{Q} = \frac{1}{\sqrt{2}}\begin{bmatrix} \widetilde{\mathbf{U}} & \widetilde{\mathbf{U}} \\ \widetilde{\mathbf{V}} & -\widetilde{\mathbf{V}} \end{bmatrix} \in \mathbb{C}^{(M+N)\times 2M}, \qquad \mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0} & -\mathbf{\Sigma} \end{bmatrix} \in \mathbb{R}^{2M\times 2M}. \tag{22}$$

If $\mathbf{A}$ is a rank-r matrix, then it is clear from (22) that $\bar{\mathbf{A}}$ has $2r$ non-zero eigenvalues that come in positive and negative pairs corresponding to the non-zero singular values of $\mathbf{A}$, and $\bar{\mathbf{A}}$ has $N + M - 2r$ zero eigenvalues. Furthermore, the eigenvectors of $\bar{\mathbf{A}}$ clearly reveal the left and right singular vectors of $\mathbf{A}$ by inspection. As a result, if the eigenvalue decomposition of $\bar{\mathbf{A}}$ given in (22) is obtained numerically, then the singular value decomposition of $\mathbf{A}$ is readily available.

In order to obtain the dominant eigenvector of $\bar{\mathbf{A}}$ (which correspond to the dominant left and right singular vectors of $\mathbf{A}$), we will consider the random component-wise updates of (2) running on the matrix $\bar{\mathbf{A}}$:

$$(\bar{\mathbf{x}}_k)_i = \begin{cases} (\bar{\mathbf{A}}\ \bar{\mathbf{x}}_{k\text{-}1})_i, & i \in \mathcal{T}_k, \\ (\bar{\mathbf{x}}_{k\text{-}1})_i, & i \notin \mathcal{T}_k, \end{cases} \tag{23}$$

where we assume that only one index is updated per iteration, i.e., $|\mathcal{T}_k| = 1$, for the sake of simplicity. The case of updating more than one index is a straightforward generalization of the approach considered here.

Let the iterand $\bar{\mathbf{x}}_k \in \mathbb{C}^{M+N}$ in (23) be partitioned as follows:

$$\bar{\mathbf{x}}_k = \begin{bmatrix} \mathbf{u}_k \\ \mathbf{v}_k \end{bmatrix}, \qquad \text{where} \qquad \mathbf{u}_k \in \mathbb{C}^M, \quad \mathbf{v}_k \in \mathbb{C}^N. \tag{24}$$

Then, the update scheme (23) can be expressed equivalently using the matrix $\mathbf{A}$ itself and the partitions $\mathbf{u}_k$ and $\mathbf{v}_k$. More precisely, we have the following:

$$\begin{aligned} &\text{If} \quad i \leqslant M, \qquad (\bar{\mathbf{x}}_k)_i = (\mathbf{u}_k)_i, \qquad \text{and} \qquad (\bar{\mathbf{A}}\ \bar{\mathbf{x}}_k)_i = \mathbf{A}_{[i,\,:]}\ \mathbf{v}_k, \\ &\text{If} \quad i > M, \qquad (\bar{\mathbf{x}}_k)_i = (\mathbf{v}_k)_{i\text{-}M}, \qquad \text{and} \qquad (\bar{\mathbf{A}}\ \bar{\mathbf{x}}_k)_i = (\mathbf{A}_{[:,\,i\text{-}M]})^{\mathrm{H}}\ \mathbf{u}_k. \end{aligned} \tag{25}$$

Due to (25) and the assumption that only one index is updated per iteration, the k^{th} iteration of (23) can be described as follows: select an index i randomly uniformly from the set $\{1\,, \cdots\,, M{+}N\}$. If the index corresponds to a row, i.e., $i \leqslant M$, then the i^{th} index of $\mathbf{u}_{k\text{-}1}$ is updated as the inner product between the i^{th} row of $\mathbf{A}$ and the vector $\mathbf{v}_{k\text{-}1}$. If the index corresponds to a column, i.e., $i > M$, then the $(i-M)^{th}$ index of $\mathbf{v}_{k\text{-}1}$ is updated as the inner product between the $(i-M)^{th}$ column of $\mathbf{A}$ and the vector $\mathbf{u}_{k\text{-}1}$. These updates are described as a pseudo-code in Algorithm 1.

Algorithm 1 Dominant Singular Vector without a Normalization Step

Assume $\|\mathbf{A}\|_2 = 1$.
Initialize $\mathbf{u} \in \mathbb{R}^M$, $\mathbf{v} \in \mathbb{R}^N$ randomly.
while convergence **do**
  $i \sim \mathcal{U}\{1, \cdots, M+N\}$
  if $i \leqslant M$ **then**
    $u_i \leftarrow \mathbf{A}_{[i,\,:]}\ \mathbf{v}$
  else
    $i \leftarrow (i-M)$
    $v_i \leftarrow \left(\mathbf{A}_{[:,\,i]}\right)^{\mathrm{H}}\ \mathbf{u}$
return $\mathbf{u}$ and $\mathbf{v}$

Algorithm 2 Dominant r Singular Vectors with a Normalization Step

Initialize $\mathbf{C}, \mathbf{U} \in \mathbb{R}^{M\times r}$, $\mathbf{V} \in \mathbb{R}^{N\times r}$ randomly.
while convergence **do**
  $i \sim \mathcal{U}\{1, \cdots, M+N\}$
  if $i \leqslant M$ **then**
    $\mathbf{C}_{[i,\,:]} \leftarrow \mathbf{A}_{[i,\,:]}\ \mathbf{V}$
    Set $\mathbf{U}$ as $\mathbf{C} = \mathbf{U}\,\mathbf{T}$ with $T_{i,i} \geqslant 0$.
  else
    $i \leftarrow (i-M)$
    $\mathbf{V}_{[i,\,:]} \leftarrow \left(\mathbf{A}_{[:,\,i]}\right)^{\mathrm{H}}\ \mathbf{U}$
return $\mathbf{U}$ and $\mathbf{V}$

Since $\bar{\mathbf{A}}$ is a normal matrix irrespective of the value of $\mathbf{A}$ and Algorithm 1 is equivalent to the update scheme (23), we can use Theorem 1 to study the convergence of the algorithm. The following theorem shows that Algorithm 1 indeed converges to the left and right dominant singular vectors of the matrix $\mathbf{A}$.

THEOREM 2. *Assume that the matrix $\mathbf{A} \in \mathbb{C}^{M\times N}$ has $\|\mathbf{A}\|_2 = 1$, i.e. the maximum singular value is unity. Assume further that the eigenspace parameter ρ of the matrix $\bar{\mathbf{A}}$ satisfies $\rho < 1$. Then, $\mathbf{u}$ and $\mathbf{v}$ in Algorithm 1 converge to the left and right dominant singular vectors of $\mathbf{A}$, respectively.*

Proof. We start by noting that Algorithm 1 is equivalent to the update scheme of (23) due to the equivalence in (25). Thus, the convergence of (23) is equivalent to the convergence of the algorithm.

Let σ_i denote the i^{th} largest singular value of $\mathbf{A}$. Since $\|\mathbf{A}\|_2 = 1$, we have $1 = \sigma_1 \geqslant \sigma_2 \geqslant \cdots \sigma_M \geqslant 0$. From (22) we have that $\bar{\mathbf{A}}$ is an Hermitian matrix with eigenvalues $\lambda(\bar{\mathbf{A}}) = \pm\,\sigma(\mathbf{A})$ with additional $N-M$ zero eigenvalues. Thus, $\bar{\mathbf{A}}$ has a unit eigenvalue, and its non-unit eigenvalues satisfy -1 $\leqslant \lambda_i < 1$.

Due to random component-wise nature of the update scheme in (23), we have $\delta_{\mathsf{T}} > 0$. In addition, we assume that the eigenspace parameter of $\bar{\mathbf{A}}$ satisfies $\rho < 1$. Thus, Corollary 1 shows that the convergence region given in (13) contains the interval [-1 1) due to the fact that $\alpha = \delta_{\mathsf{T}}\,(\rho - 1) < 0$. Since all non-unit eigenvalues of $\bar{\mathbf{A}}$ satisfy -1 $\leqslant \lambda_i < 1$, Corollary 1 guarantees that the iterand $\bar{\mathbf{x}}_k$ in (23) converges to an eigenvector of $\bar{\mathbf{A}}$ with the eigenvalue 1. Then, it is clear from (22) that the partitions of $\bar{\mathbf{x}}_k$ defined in (24) converge to the left and right

singular vectors of $\mathbf{A}$ corresponding to the singular value 1. That is, $\mathbf{u}_k$ converges to the left dominant singular vector $\mathbf{A}$, and $\mathbf{v}_k$ converges to the right dominant singular vector of $\mathbf{A}$. □

The convergence guarantee provided by Theorem 2 is based on two assumptions. The first one, $\|\mathbf{A}\|_2 = 1$, is in fact a necessary condition for the convergence of the updates in (23). Existence of a singular value 1 implies the existence of a fixed point of the component-wise updates.[7] The second one, which requires the eigenspace parameter to satisfy $\rho < 1$, is a technical assumption that is needed to prove the convergence precisely. It is not a necessary condition, and it is in fact satisfied in many practical applications. It is also important to note that Theorem 2 does not assume realness of the matrix $\mathbf{A}$. The algorithm is guaranteed to converge even when $\mathbf{A}$ is a complex matrix, in which case dominant singular vectors are complex as well.

3.1 Rank-r Approximation of an Arbitrary Matrix

Although the assumption $\|\mathbf{A}\|_2 = 1$ required by Algorithm 1 can be satisfied easily by normalizing the data matrix $\mathbf{A}$ with its largest singular value, the computation of the largest singular value itself may not be very practical especially when $\mathbf{A}$ is large in dimensions. It is, in fact, possible to remove this assumption by introducing a normalization step into the algorithm. Furthermore, it is also possible to extend Algorithm 1 in such a way that it converges to the dominant r singular vectors of $\mathbf{A}$ together with the top-r singular values for an arbitrary value of r. The extended version of the algorithm is presented in Algorithm 2.

Algorithm 2 differs from Algorithm 1 in three ways: Firstly, the vector variables $\mathbf{u}$ and $\mathbf{v}$ in Algorithm 1 are extended to be matrices with r columns. Secondly, Algorithm 2 uses an auxiliary variable $\mathbf{C}$. Thirdly, and the most importantly, Algorithm 2 uses a *QR decomposition* (Line 6) that serves as the normalization step. More precisely, instead of updating the variable $\mathbf{U}$ directly, Algorithm 2 first updates the auxiliary variable $\mathbf{C}$ (Line 5), and then updates $\mathbf{U}$ as the unitary part of the QR decomposition of $\mathbf{C}$. We note that the matrix $\mathbf{T}$ in Line 6 of the algorithm denotes the upper-triangular part of the QR decomposition of $\mathbf{C}$. Without loss of generality, it is assumed that $\mathbf{T}$ has non-negative diagonal entries, and its diagonal entries are in the descending order.

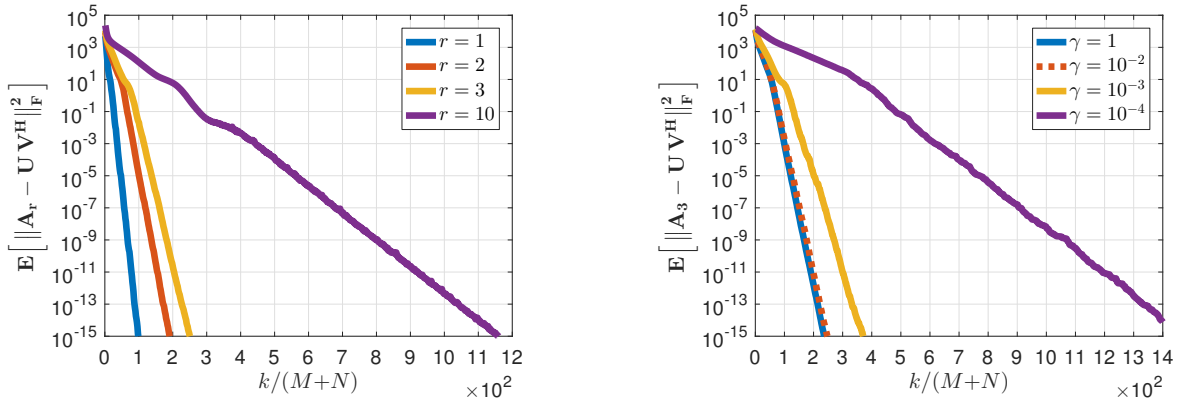
Although the convergence of Algorithm 1 is ensured by Theorem 2, we do not provide an explicit proof for the convergence of Algorithm 2. Nevertheless, by the virtue of Theorem 2 we can argue for the convergence of Algorithm 2 since it is a natural extension of Algorithm 1 with an additional normalization step. We observe that the variables of Algorithm 2 converge as follows:

$$\mathbf{U} \to \tilde{\mathbf{U}}_r, \qquad \mathbf{T} \to \mathbf{\Sigma}_r^2, \qquad \mathbf{V} \to \tilde{\mathbf{V}}_r\, \mathbf{\Sigma}_r, \tag{26}$$

where $\tilde{\mathbf{U}}_r$ and $\tilde{\mathbf{V}}_r$ are the first r columns of $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$, respectively, and $\mathbf{\Sigma}_r$ is the top-left $r \times r$ block of $\mathbf{\Sigma}$. Thus, the product $\mathbf{U}\,\mathbf{V}^{\mathrm{H}}$ converges to $\mathbf{A}_r$, which is the best rank-r approximation of $\mathbf{A}$, i.e, $\mathbf{A}_r = \tilde{\mathbf{U}}_r\, \mathbf{\Sigma}_r\, \tilde{\mathbf{V}}_r^{\mathrm{H}}$.

In order to verify its convergence, we simulate Algorithm 2 on the test matrix MEDLINE,[27] which is a full-rank and sparse matrix of size 1033×5735. We measure the convergence of the algorithm in terms of the squared Frobenius norm of the difference between $\mathbf{A}_r$ and the product $\mathbf{U}\,\mathbf{V}^{\mathrm{H}}$. Since the update index is selected randomly (Line 3) in every iteration of Algorithm 2, the error term, i.e, $\|\mathbf{A}_r - \mathbf{U}\,\mathbf{V}^{\mathrm{H}}\|_{\mathrm{F}}^2$, is a random variable as well. So, we compute the expected error by averaging over 10^3 independent runs of the algorithm. These results are presented in Figure 4(a) for the cases of $r \in \{1, 2, 3, 10\}$, which numerically verify the convergence of the algorithm. Note that the algorithm requires more iterations to converge as the value of r gets larger.

We note that Line 5 of Algorithm 2 updates only a row of the auxiliary variable $\mathbf{C}$ in every iteration. Since the matrix $\mathbf{C}$ is not expected to change significantly during an iteration, the normalization step in Line 6 can be skipped in some iterations in order to reduce the overall computational complexity of the algorithm. In order to verify this claim, we modify the implementation of the algorithm in such a way that Line 6 is executed with probability γ. So, the modified implementation reduces to Algorithm 2 when $\gamma = 1$. For the case of $r = 3$, we compute the expected error of the modified implementation by averaging over 10^3 independent runs. These results are presented in Figure 4(b) for the values of $\gamma \in \{1, 10^{-2}, 10^{-3}, 10^{-4}\}$, which shows that the modified implementation keeps converging for wide range of values of γ. More interestingly, the rate of convergence remains visually the same even when the normalization step is executed with probability as low as $\gamma = 10^{-2}$. Moreover, the rate of convergence decreases marginally when $\gamma = 10^{-3} \approx 1/M$. This is consistent with the fact



(a) (b)

Figure 4. (a) Convergence of Algorithm 2 for various different values of r. (b) Convergence of Algorithm 2 for the case $r = 3$ when the normalization step in Line 6 is executed with probability γ. Here k indicates the number of iterations.

that an iteration of Algorithm 2 updates only one row of $\mathbf{C}$ that has M rows in total. Nevertheless, when γ has a very small value, e.g., $\gamma = 10^{-4}$, the algorithm indeed gets significantly slower.

Regarding the computational complexity of the algorithm, note that the cost of Line 5, Line 6 and Line 9 are $\mathcal{O}(Nr)$, $\mathcal{O}(Mr^2)$, and $\mathcal{O}(Mr)$, respectively. However, the algorithm gets to Lines 5 and 6 with probability $M/(M+N)$, and it gets to Line 9 with probability $N/(M+N)$. When we further assume that Line 6 is executed with probability γ, *the average cost* of an iteration of Algorithm 2 can be found as follows:

$$\mathbb{E}[\text{computatinal cost per iteration}] = \mathcal{O}\left(\frac{MNr + \gamma\, M^2\, r^2}{M+N}\right) \approx \mathcal{O}\left(\frac{MNr}{M+N}\right), \tag{27}$$

where the approximation is valid when $\gamma \leqslant N/(Mr)$, which is acceptable in practice as suggested by Figure 4(b). On the other hand, the synchronous form of (23) requires $\mathcal{O}(MNr)$ multiplications per iteration. In order to compensate the additional factor of $M+N$, the iteration index k is normalized by $M+N$ in both Figures 4(a) and 4(b).

Relevance of Algorithm 2 follows from its applicability for asynchronous and distributed implementation. Since a single iteration of the algorithm requires a partial information of the matrix $\mathbf{A}$, (i.e., a single column or row) multiple processors can operate on the same matrix $\mathbf{A}$ simultaneously without requiring any ordering among them. More importantly, it is possible to extend Algorithm 2 in such a way that the data matrix $\mathbf{A}$ is partitioned into multiple smaller pieces, and each piece is stored in a different processing core as we discuss next.

3.2 Distributed Implementation with Partial Data Storage

In this section we will assume that the matrix $\mathbf{A} \in \mathbb{C}^{M\times N}$ represents a collection of data points where each column of the matrix is a data point in the M-dimensional feature space, and $\mathbf{A}$ has N data points in total. In some of the applications the number of the data points, N, can be too large for $\mathbf{A}$ to be stored in a single core. Thus, the data needs to be partitioned into smaller collections and stored in different cores. It could also be the case that the data is already located in different places and may not be available directly due to privacy concerns. The asynchronous (component-wise) nature of Algorithm 2 makes it suitable to compute the dominant singular vectors of $\mathbf{A}$ in these scenarios. Although Algorithm 2 is not directly applicable, it can be modified to handle these scenarios as well. For this purpose assume that the matrix $\mathbf{A}$ is partitioned into P blocks as follows:

$$\mathbf{A} = [\mathbf{A}^{(1)} \quad \mathbf{A}^{(2)} \quad \cdots \quad \mathbf{A}^{(P)}], \tag{28}$$

where $\mathbf{A}^{(p)} \in \mathbb{C}^{M\times N_p}$ denotes the p^{th} partition holding corresponding N_p data points, so $\sum_{p=1}^{P} N_p = N$.

When Algorithm 2 is utilized on the partitioned data, the column selection phase can be done in a straightforward manner since columns of $\mathbf{A}$ are assumed to be the individual data points. On the contrary, the row selection

phase of Algorithm 2 (Line 5) requires to access the same row of *all the partitions*. Due to the distributed nature of the data it may not be possible for a processor to access all the partitions simultaneously. Nevertheless, the required inner product can be written as a sum of partial inner products, which then can be computed locally. For this purpose, we first partition the variable $\mathbf{V} \in \mathbb{C}^{N \times r}$ in Algorithm 2 with respect to the partitions given in (28). More precisely, assume that the p^{th} core holds a *local* variable $\mathbf{V}^{(p)} \in \mathbb{C}^{N_p \times r}$. Then, the inner product involving the rows of $\mathbf{A}$ can be written as follows:

$$\mathbf{A}_{[i,:]}\, \mathbf{V} = \sum_{p=1}^{P} \mathbf{A}^{(p)}_{[i,:]}\, \mathbf{V}^{(p)}. \tag{29}$$

Notice that the quantity $\mathbf{A}^{(p)}_{[i,:]}\, \mathbf{V}^{(p)}$ can be computed in the p^{th} core locally. Once it is obtained, it can be sent to a fusion center in order to update the value of the variable $\mathbf{U}$. Based on this observation, we propose Algorithm 3 for the computation of the dominant singular vectors for the case of distributed data.

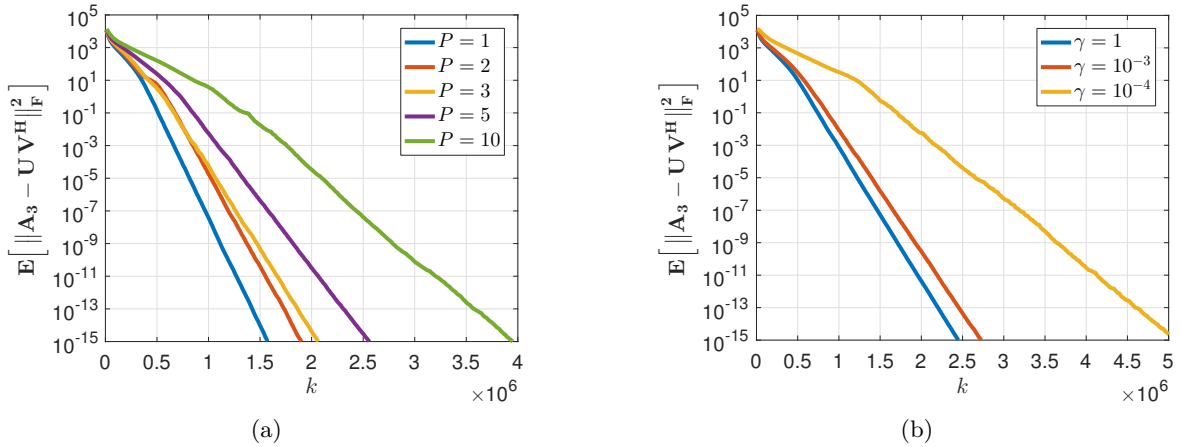
Algorithm 3 Distributed Computation of Dominant r Singular Vectors

1: Initialize $\mathbf{C} \in \mathbb{R}^{M \times r \times P}$, $\mathbf{U} \in \mathbb{R}^{M \times r}$, $\mathbf{V}^{(1)} \in \mathbb{R}^{N_1 \times r}, \cdots, \mathbf{V}^{(P)} \in \mathbb{R}^{N_P \times r}$ randomly.
2: **while** convergence **do**
3: $\quad p \sim \mathcal{U}\{1, \cdots, P\}$ ▷ Select a partition randomly
4: $\quad i \sim \mathcal{U}\{1, \cdots, M + N_p\}$ ▷ Select an index randomly
5: $\quad$ **if** $i \leqslant M$ **then**
6: $\quad\quad \mathbf{C}_{[i,:,p]} \leftarrow \mathbf{A}^{(p)}_{[i,:]}\, \mathbf{V}^{(p)}$ ▷ Partial inner product
7: $\quad\quad$ Compute $\widehat{\mathbf{C}}$ as $\widehat{C}_{i,j} = \sum_{p=1}^{P} C_{i,j,p}$ ▷ Data fusion
8: $\quad\quad$ Set $\mathbf{U}$ as $\widehat{\mathbf{C}} = \mathbf{U}\,\mathbf{T}$ with $T_{i,i} \geqslant 0$. ▷ QR decomposition
9: $\quad$ **else**
10: $\quad\quad i \leftarrow (i - M)$
11: $\quad\quad \mathbf{V}^{(p)}_{[i,:]} \leftarrow \left(\mathbf{A}^{(p)}_{[:,i]}\right)^{\mathrm{H}} \mathbf{U}$ ▷ Update the local variable

It is important to note that Algorithm 3 can be implemented in a distributed manner with the help of a data fusion center and a shared memory. In such implementation the variable $\mathbf{C}$ corresponds to the data collected by the fusion center, and the variable $\mathbf{U}$ is assumed to be in a shared memory. On the contrary, the data matrix $\mathbf{A}^{(p)}$ and the variable $\mathbf{V}^{(p)}$ are stored in the p^{th} processor locally. Then, Algorithm 3 proceeds as follows: it first selects the p^{th} partition randomly uniformly among all P partitions (Line 3). Then, the processor randomly selects and performs either one of the following two actions: 1) It uses the local variable $\mathbf{V}^{(p)}$, computes a partial inner product, and sends it to the fusion center (Line 6). 2) It uses the global shared variable $\mathbf{U}$ in order to update the local variable $\mathbf{V}^{(p)}$ (Line 11). In the mean-time, whenever the fusion center is updated, it first computes the full inner products (Line 7), and then updates the global variable $\mathbf{U}$ via QR decomposition (Line 8). We also note that in the case of a single partition, i.e., $P = 1$, Algorithm 3 reduces to Algorithm 2.

Algorithm 3 has two major benefits. First, the p^{th} processor uses the p^{th} data partition only, thus all the computations can be done locally on a processor. More importantly, the data itself is never shared: a summary of the data (in the form an inner product) is sent to the fusion center. Only shared information is the variable $\mathbf{U}$, which converges to the dominant r left singular vectors of the whole data matrix $\mathbf{A}$. Secondly, due to the randomized nature of the algorithm, a synchronization between the processors is not required. The processors are allowed to interact with the fusion center independently at any order.

Similar to Algorithm 2, the variable $\mathbf{U}$ of Algorithm 3 converges to $\widetilde{\mathbf{U}}_r$, and the variables $\mathbf{V}^{(p)}$ converge to the corresponding partitions of $\widetilde{\mathbf{V}}_r\, \mathbf{\Sigma}_r$ similar to (26). In order to verify the convergence of Algorithm 3 numerically, we use the test matrix MEDLINE[27] and divide it into P blocks as in (28), where each partition has



(a) (b)

Figure 5. (a) Convergence of Algorithm 3 for the case of $r = 3$, i.e., rank-3 approximation, for different numbers of P data partitions. (b) Convergence of Algorithm 3 for the case $r = 3$ with $P = 5$ partitions when the normalization step in Line 8 is executed with probability γ. Here k indicates the number of iterations.

size (approximately) N/P. We consider the case of $r = 3$, i.e., rank-3 approximation, and the error is computed as the expected value of $\|\mathbf{A}_3 - \mathbf{U}\,\mathbf{V}^{\mathrm{H}}\|_{\mathrm{F}}^2$ where $\mathbf{V}$ denotes the matrix constructed by cascading the local variables $\mathbf{V}^{(p)}$. The expected error is computed by averaging over 10^3 independent runs of the algorithm, and the results are presented in Figure 5(a) for the cases of $P \in \{1, 2, 3, 5, 10\}$, which verify the convergence numerically.

In order to demonstrate the robustness of Algorithm 3 to stale data in the case of distributed implementation, we modify the algorithm in such a way that the normalization step in Line 8 is executed with probability γ. Thus, the processors do not always use the value of $\mathbf{U}$ that corresponds to the most recent value of $\mathbf{C}$; rather, they use an outdated value of $\mathbf{U}$. The modified implementation is simulated for the case of $r = 3$ with $P = 5$ partitions, and the error is computed as before by averaging over 10^3 independent runs. The simulation results are presented in Figure 5(b) for the values of $\gamma \in \{1, 10^{-3}, 10^{-4}\}$. It is clear from the figure that even when the shared memory (the variable $\mathbf{U}$) is updated with probability as low as $\gamma = 10^{-3} \approx 1/M$, the modified implementation continues to converge as fast as Algorithm 3 itself, which indicates the robustness of the algorithm to the use of stale data.

4. CONCLUSION

In this paper we studied a random component-wise variant of the regular power iteration. We showed that the randomized updates have significantly different convergence characteristics. In particular, we proved that random updates converge in the mean-squared sense to an eigenvector of the eigenvalue 1 even when the matrix has a spectral radius larger than unity. We also demonstrated and discussed how the sign (or, the phase when complex) of an eigenvalue affects the rate of convergence of random component-wise updates. As an application, we reformulated the component-wise power iteration in order to compute the dominant singular vectors of a given data matrix. The proposed approach is proven to converge when computing the rank-1 approximation of a normalized data matrix, and its convergence is verified numerically for an arbitrary rank approximation of an arbitrary data matrix. The proposed algorithm is extended in order to handle large-scale distributed data with distributed asynchronous computation. The convergence of the extended algorithm is verified numerically for various different number of data partitions.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Dimitri Van De Ville for the invitation to write this article.

APPENDIX A. PROOF OF LEMMA 1

We note that $\mathbf{U} \in \mathbb{C}^{N \times (N-m)}$ has orthonormal columns, i.e., $\mathbf{U}^{\mathrm{H}}\,\mathbf{U} = \mathbf{I}$ and prove the upper bound in (8) first. Note that $\mathbf{U}\,\mathbf{U}^{\mathrm{H}} \preceq \mathbf{I}$. Then we can write the following:

$$(\mathbf{U}\,\mathbf{U}^{\mathrm{H}})_{i,i} = \mathbf{e}_i^{\mathrm{H}}\,\mathbf{U}\,\mathbf{U}^{\mathrm{H}}\,\mathbf{e}_i \leqslant \mathbf{e}_i^{\mathrm{H}}\mathbf{e}_i = 1 \quad \Longrightarrow \quad \mathrm{diag}\left(\mathbf{U}\,\mathbf{U}^{\mathrm{H}}\right) \preceq \mathbf{I} \quad \Longrightarrow \quad \mathbf{U}^{\mathrm{H}}\,\mathrm{diag}\left(\mathbf{U}\,\mathbf{U}^{\mathrm{H}}\right)\mathbf{U} \preceq \mathbf{I} \tag{30}$$

where $\mathbf{e}_i$ denotes the i^{th} column of the identity matrix of dimension N.

We now prove the lower bound in (8). Let $\mathbf{u}_i$ denote the i^{th} row of $\mathbf{U}$, then it is clear that $(\mathbf{U}\,\mathbf{U}^{\mathrm{H}})_{i,j} = \mathbf{u}_i\,\mathbf{u}_j^{\mathrm{H}}$. Let $\mathbf{x} \in \mathbb{C}^N$ be an arbitrary vector. Then,

$$\mathbf{x}^{\mathrm{H}}\,\mathbf{U}\,\mathbf{U}^{\mathrm{H}}\,\mathbf{x} = \left|\mathbf{x}^{\mathrm{H}}\,\mathbf{U}\,\mathbf{U}^{\mathrm{H}}\,\mathbf{x}\right| = \left|\sum_{i=1}^{N}\sum_{j=1}^{N} x_i^*\,(\mathbf{U}\,\mathbf{U}^{\mathrm{H}})_{i,j}\,x_j\right| = \left|\sum_{i=1}^{N}\sum_{j=1}^{N} x_i^*\,\mathbf{u}_i\,\mathbf{u}_j^{\mathrm{H}}\,x_j\right| \leqslant \sum_{i=1}^{N}\sum_{j=1}^{N} |x_i|\,\left|\mathbf{u}_i\,\mathbf{u}_j^{\mathrm{H}}\right|\,|x_j| \tag{31}$$

$$\leqslant \sum_{i=1}^{N}\sum_{j=1}^{N} |x_i|\,\|\mathbf{u}_i\|_2\|\mathbf{u}_j\|_2\,|x_j| = \left(\sum_{i=1}^{N} |x_i|\,\|\mathbf{u}_i\|_2\right)^2 \leqslant N\sum_{i=1}^{N} |x_i|^2\,\|\mathbf{u}_i\|_2^2 = N\,\mathbf{x}^{\mathrm{H}}\,\mathrm{diag}\left(\mathbf{U}\,\mathbf{U}^{\mathrm{H}}\right)\mathbf{x}. \tag{32}$$

Then, the inequity (32) implies that

$$\mathbf{U}\,\mathbf{U}^{\mathrm{H}} \preceq N\,\mathrm{diag}\left(\mathbf{U}\,\mathbf{U}^{\mathrm{H}}\right) \qquad \Longrightarrow \qquad \mathbf{U}^{\mathrm{H}}\mathbf{U}\,\mathbf{U}^{\mathrm{H}}\,\mathbf{U} \;\preceq\; N\,\mathbf{U}^{\mathrm{H}}\,\mathrm{diag}\left(\mathbf{U}\,\mathbf{U}^{\mathrm{H}}\right)\mathbf{U}, \tag{33}$$

which proves the lower bound due to the fact that $\mathbf{U}^{\mathrm{H}}\,\mathbf{U} = \mathbf{I}$.

REFERENCES

[1] Parlett, B. N., [*The Symmetric Eigenvalue Problem (Classics in Applied Mathematics)*], Society for Industrial and Applied Mathematics (1987).

[2] Halko, N., Martinsson, P. G., and Tropp, J. A., "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review* **53**(2), 217–288 (2011).

[3] Yuan, X.-T. and Zhang, T., "Truncated power method for sparse eigenvalue problems," *Journal of Machine Learning Research* **14**(1), 899–925 (2013).

[4] Hardt, M. and Price, E., "The noisy power method: A meta algorithm with applications," in [*Advances in Neural Information Processing Systems 27*], 2861–2869 (2014).

[5] Journe, M., Nesterov, Y., Richtrik, P., and Sepulchre, R., "Generalized power method for sparse principal component analysis," *The Journal of Machine Learning Research* **11**, 517–553 (2010).

[6] Page, L., Brin, S., Motwani, R., and Winograd, T., "The pagerank citation ranking: Bringing order to the web.," technical report, Stanford InfoLab (November 1999).

[7] Teke, O. and Vaidyanathan, P. P., "Random node-asynchronous updates on graphs," *IEEE Trans. Signal Process.* **67**, 2794–2809 (June 2019).

[8] Teke, O. and Vaidyanathan, P. P., "Asynchronous nonlinear updates on graphs," in [*Asilomar Conf. on Signals, Systems and Computers*], 998–1002 (Oct. 2018).

[9] Teke, O. and Vaidyanathan, P. P., "Node-asynchronous implementation of rational filters on graphs," in [*Proc. Int. Conf. Acoust. Speech, Signal Process. (ICASSP)*], 7530–7534 (May 2019).

[10] Kaczmarz, S., "Angenaherte auflosung von systemen linearer gleichungen," *Bull. Internat. Acad. Polon. Sci. Letters A* , 335–357 (1937).

[11] Wright, S. J., "Coordinate descent algorithms," *Mathematical Programming* **151**, 3–34 (Jun 2015).

[12] Wang, J., Wang, W., Garber, D., and Srebro, N., "Efficient coordinate-wise leading eigenvector computation," *arXiv* **1702.07834** (2017).

[13] Beck, A. and Tetruashvili, L., "On the convergence of block coordinate descent type methods," *SIAM Journal on Optimization* **23**(4), 2037–2060 (2013).

[14] Lei, Q., Zhong, K., and Dhillon, I. S., "Coordinate-wise power method," in [*Advances in Neural Inf. Process. Systems (NIPS)*], 2064–2072 (2016).

[15] Chazan, D. and Miranker, W., "Chaotic relaxation," *Linear Algebra and its Applications* **2**, 199–222 (Apr. 1969).
[16] Baudet, G. M., "Asynchronous iterative methods for multiprocessors," *J. ACM* **25**, 226–244 (Apr. 1978).
[17] Bertsekas, D. P., "Distributed asynchronous computation of fixed points," *Mathematical Programming* **27**, 107–120 (Sep. 1983).
[18] Avron, H., Druinsky, A., and Gupta, A., "Revisiting asynchronous linear solvers: Provable convergence rate through randomization," *J. ACM* **62**, 51:1–51:27 (Dec. 2015).
[19] Peng, Z., Xu, Y., Yan, M., and Yin, W., "Arock: An algorithmic framework for asynchronous parallel coordinate updates," *SIAM Journal on Scientific Computing* **38**(5), A2851–A2879 (2016).
[20] Teke, O. and Vaidyanathan, P. P., "IIR filtering on graphs with random node-asynchronous updates," *Submitted to IEEE TSP* (June 2019).
[21] Teke, O. and Vaidyanathan, P. P., "The asynchronous power iteration: A graph signal perspective," in [*Proc. Int. Conf. Acoust. Speech, Signal Process. (ICASSP)*], 4059–4063 (Apr. 2018).
[22] Hardt, M. and Roth, A., "Beyond worst-case analysis in private singular vector computation," in [*Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*], 331–340 (2013).
[23] Shamir, O., "A stochastic pca and svd algorithm with an exponential convergence rate," in [*Proc. of the 32st Int. Conf. Machine Learning (ICML 2015)*], 144–152 (2015).
[24] Shamir, O., "Fast stochastic algorithms for svd and pca: Convergence properties and convexity," in [*Proc. of the 33st Int. Conf. Machine Learning (ICML 2016)*], 248–256 (2016).
[25] Warmuth, M. K. and Kuzmin, D., "Randomized online pca algorithms with regret bounds that are logarithmic in the dimension," *The Journal of Machine Learning Research* **9**, 2287–2320 (2008).
[26] Mitliagkas, I., Caramanis, C., and Jain, P., "Memory limited, streaming pca," in [*Advances in Neural Information Processing Systems 26*], 2886–2894 (2013).
[27] Text to Matrix Generator, "http://scgroup20.ceid.upatras.gr:8000/tmg/," (2019).